Informatics 2D: Reasoning and Agents

Alex Lascarides

School of informatics

The University of Edinburgh

Lecture 16b: Representing Planning Problems with PDDL

Where are we?

Last time. . .

- Why we need a formal representation of planning problems:
  - states, goals, actions

  that supports efficient algorithms for finding a valid plan

Now: Representing planning problems

- Planning Domain Definition Language (PDDL)

Later: The planning algorithms

Representing planning problems

- Need a language expressive enough to cover interesting problems, restrictive enough to allow efficient algorithms.
- Planning Domain Definition Language or PDDL
- PDDL will allow you to express:
  1. states
  2. actions: a description of transitions between states
  3. and goals: a (partial) description of a state.

Representing States and Goals in PDDL

- **States** represented as conjunctions of propositional or function-free first order positive literals:
  - $Happy \wedge Sunshine$,
    $At(Plane_1, Melbourne) \wedge At(Plane_2, Sydney)$
- So these aren't states:
  - $At(x, y)$ (no variables allowed),
    $Love(Father(Fred), Fred)$ (no function symbols allowed)
    $\neg Happy$ (no negation allowed).

  Closed-world assumption!
- A **goal** is a partial description of a state, and you can use negation, variables etc. to express that description.
  - $\neg Happy$, $At(x, SFO)$, $Love(Father(Fred), Fred)$ ...

Actions in PDDL

$Action(Fly(p, from, to),$

$\quad$ Precond: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$

$\quad$ Effect: $\neg At(p, from) \land At(p, to))$

- Actually **action schemata**, as they may contain variables
- Action name and parameter list serves to identify the action
- **Precondition**: defines states in which action is **executable**:
  - Conjunction of positive and negative literals, where all variables must occur in action name.
- **Effect**: defines how literals in the input state get changed (anything not mentioned stays the same).
  - Conjunction of positive and negative literals, with all its variables also in the preconditions.
  - Often positive and negative effects are divided into **add list** and **delete list**

The semantics of PDDL: States and their Descriptions

- $s \models At(P_1, SFO)$ iff $At(P_1, SFO) \in s$
  $s \models \neg At(P_1, SFO)$ iff $At(P_1, SFO) \notin s$
  $s \models \phi(x)$ iff there is a ground term d such that $s \models \phi[x/d]$.
  $s \models \phi \wedge \psi$ iff $s \models \phi$ and $s \models \psi$

The Semantics of PDDL: Applicable Actions

- Any action is **applicable** in any state that satisfies the precondition with an appropriate substitution for parameters.
- Example: State

$$At(P_1, Melbourne) \wedge At(P_2, Sydney) \wedge Plane(P_1) \wedge Plane(P_2)$$
$$\wedge Airport(Sydney) \wedge Airport(Melbourne) \wedge Airport(Heathrow)$$

satisfies

$$At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$$

with substitution (among others)

$$\{p/P_2, from/Sydney, to/Heathrow\}$$

The semantics of PDDL: The Result of an Action

- **Result** of executing action a in state s is state s' with any positive literal P in a's Effects added to the state and every negative literal $\neg P$ removed from it (under the given substitution) .
- In our example s' would be

$$At(P_1, Melbourne) \wedge At(P_2, Heathrow) \wedge Plane(P_1) \wedge Plane(P_2) \\ \wedge Airport(Sydney) \wedge Airport(Melbourne) \wedge Airport(Heathrow)$$

- "PDDL assumption": every literal not mentioned in the effect remains unchanged (cf. frame problem)
- **Solution** = action sequence that leads from the initial state to a state that satisfies the goal.

Summary

- Introduced PDDL:
  - states, goals, actions
- Fragment of first order logic
- Restrictive enough to design efficient algorithms for solving planning problems
  - Given the current state and goal, find a sequence of actions to get from the former to the latter.

Next Time

- Some simple examples of planning problems